Web Services – Not Just For Business Anymore

CSC Seminar
March 23, 2006
Dave Stampf

Outline

- Web Services Definition
- Fundamental Technologies
  - Http (brief)
  - XML (briefer)
  - Soap (briefest)
- Examples
  - Developing a Web Service from Scratch
  - Google
  - Atlas
  - KEGG
- You & BNL as a participant in this “web”

Where we are going...

- A Web Service is a piece of “<business> logic” on the Internet accessible via standard Internet protocols such as HTTP
  - XML Based (open and visible – if not encrypted) to all)
  - Loosely coupled (not Corba)
  - Course grained (logical operations are complex)
  - Synchronous or Asynchronous
  - Supports RPC (Almost invisible to programmers)
  - Supports document exchange (Big data)

Web Services vs CGI

- First, note that most businesses prefer Web Services to CGI
  - Development model is more familiar & can easily use legacy code
  - Performance – generally ws considered superior but YMMV
  - Security – Web Services are generally more secure – sandboxes, secure loaders, encryption, etc.
  - Web Services have fewer bugs – similar to catching errors at compile time.
- Most importantly – it formalizes the relationship between the server and client (more below).

Fundamental Technology - HTTP

- HTTP network protocol is about 18 years old!
  - Free
  - Extremely Simple
  - Atomic
  - Easily extensible (e.g., services rather than pages)
  - (Are there any security people in the room?)
- Simplicity
  - I request something – you give me something
  - The End

HTTP Request

- A browser or any piece of software sends to the HTTP server a text request that looks like: (Get can be replaced by POST, PUT, ...)

```
GET index.html HTTP/1.1
Host: ...
User-Agent: ...
Accept:...
Accept-Language: ...
Accept-Encoding: ...
Accept-Charset: ...
Keep-Alive: ...
Connection: keep-alive
Other-Stuff: ...

More stuff - completely unspecified by HTTP as to its
content and/or format
```

HTTP Response

- The response is almost a reflection of the request:

```
HTTP/1.1 200 OK
Set-Cookie: ...
Content length: ...
Content-Type: text/html
Server: ...
Other header fields

More stuff sent back whose format and content are
unspecified by the HTTP standard, but is hinted at by
the Content-Type field.
```

The Request “1/2” of a Browser

```
import java.io.*;
import java.net.*;

public class SimpleHttpDemo {

    public static void main(String[] args) throws Exception {
        Socket s = new Socket("www.bnl.gov",80);
        InputStream is = s.getInputStream();
        PrintStream ps = new PrintStream(s.getOutputStream());

        ps.println("GET /world/ HTTP/1.1");
        ps.println("Host: www.bnl.gov");
        ps.println();

        int n;
        while ((n = is.read()) >= 0) {
            System.out.print((char)n);
        }
    }
}
```

XML

- XML (eXtensible Markup Language) is a W3C endorsed standard for document markup.
  - HTML was a very small subset of SGML – basically, they threw out 99% of SGML.
  - XML adds some of the complexity (and capability) of SGML back into HTML.
  - **It is so well defined and structured that computer programs can produce, parse, and transform XML without human intervention.**
- 'Nuf said – we will see some XML all too soon.

On top of XML - SOAP

- Ansi characters : email as XML : SOAP
  - SOAP (the acronym is pretty meaningless – the “S” stands for simple - hah!) specifies a way of formatting XML so that it looks a bit like a mail message – envelope, body, mime data, etc.
  - Main purpose is to organize & describe data
- Since anything can be in the payload of HTTP Request and Response packets, and since SOAP is “anything” ...
- Web services provide business logic where the request & response are SOAP encapsulated data and instructions

Revisiting the Definition - 1

- A Web Service is a piece of “<business> logic” on the Internet accessible via standard Internet protocols such as HTTP
  - XML Based (open and visible to all) –
    - specific requests are made to the service, encapsulating the request and parameters in an XML document within the HTTP request.
    - Responses are likewise presented to the requester, encapsulating the response in an XML document within the HTTP response.
- A note on Atlas' PANDA

Revisiting the Definition - 2

- A Web Service is a piece of “<business> logic” on the Internet accessible via standard Internet protocols such as HTTP
  - Loosely coupled (not Corba)
    - Web services do not provide network objects – they provide methods/subroutines.
    - You not only don't care *how* the request is satisfied, you probably don't even care *where* it is satisfied – or how much help the server received from other services.
    - Implementation code can be modified/updated/improved or otherwise changed independently of the clients (and vice versa)

Revisiting the Definition - 3

- A Web Service is a piece of “<business> logic” on the Internet accessible via standard Internet protocols such as HTTP
  - Course grained (logical operations are complex)
    - Packaging up XML, transmitting it, having the web server decide who should do it, sending the info to that bit of code, *, reformatting the response in XML and sending it back takes *time.*
    - To make this worthwhile many (10^6 ?) instructions should do some real work for each request – that is, it should not be a simple data operation
    - You may want to consider encapsulating database queries. (e.g. Amazon)

Revisiting the Definition - 4

- A Web Service is a piece of “<business> logic” on the Internet accessible via standard Internet protocols such as HTTP
  - Supports RPC (Almost invisible to programmers)
    - A programmer on the server side wishes to make available an existing method/function to the world. She writes or gets the function and “pushes a button”.
    - A programmer on the server side wishes to use a service and would prefer that it look like a library function. He links with the library, calls the function & gets an answer.
  - We'd like to have the “button to push” and the library to connect to be as unobtrusive as possible.

One last thought on the definition...

- The use of http, xml & loose coupling enable the use of cross platform and cross computer language interoperability.
  - Java, C, C++, Perl, Ruby, Python, Php, C#, VB, Matlab ... all can act as the server or client.
- Performance may be an issue, but this can sometimes be addressed by getting one's hands dirty in XML & SOAP. (e.g., you can add attachments – bit maps – to the request/response to avoid encoding delays).

Creating and Consuming a Web Service

- If you read most any book on creating web services, it seems way too complex. I'd suggest you start with a modern up-to-date IDE.
- The (very simple) example is in Java, but only 2 lines of Java are ever written! Everything else is handled within the development environment – Netbeans – which an incredibly powerful and free tool.
- So, to the demo...

Steps to Define a Service

- Build the interface (Method signature – no details)
- Implement the interface (details)
- Deploy the service (convert the interface to a “WSDL”, and download code to a server.
- Register
- Sit back

Steps to Create a Client

- Download and ingest the WSDL (produces a library)
- Link the library with your code.
- Call the function.

Example - Google

- Increasingly, commercial operations (Google, Amazon, etc.) are providing access to their internals via web services as well as web pages.
  - “ With the Google Web APIs service, software developers can query more than 8 billion web pages directly from their own computer programs. The Google web search API uses the SOAP and WSDL standards.”
  - See http://code.google.com/apis.html for lots more info and directions on how you can do it.

Using the Google API

```
package gov.bnl.csc.drs;

import java.io.*;
import com.google.soap.search.*;

public class GoogleWebServiceExample {

    public static void main(String[] args) throws Exception {

        GoogleSearch s = new GoogleSearch(); // create search, set key
        s.setKey("keJivdQFHBWsm9DvHCRghWyFhov4");
        s.setQueryString(“RHIC”);

        GoogleSearchResult r = s.doSearch();
        System.out.println("Google Search Results:");
        GoogleSearchResultElement[] elements = r.getResultElements();
        for (int i = 0; i < elements.length; i++) {
            System.out.println(elements[i].getURL() + " " +
                               elements[i].getTitle());
        }
    }
}
```

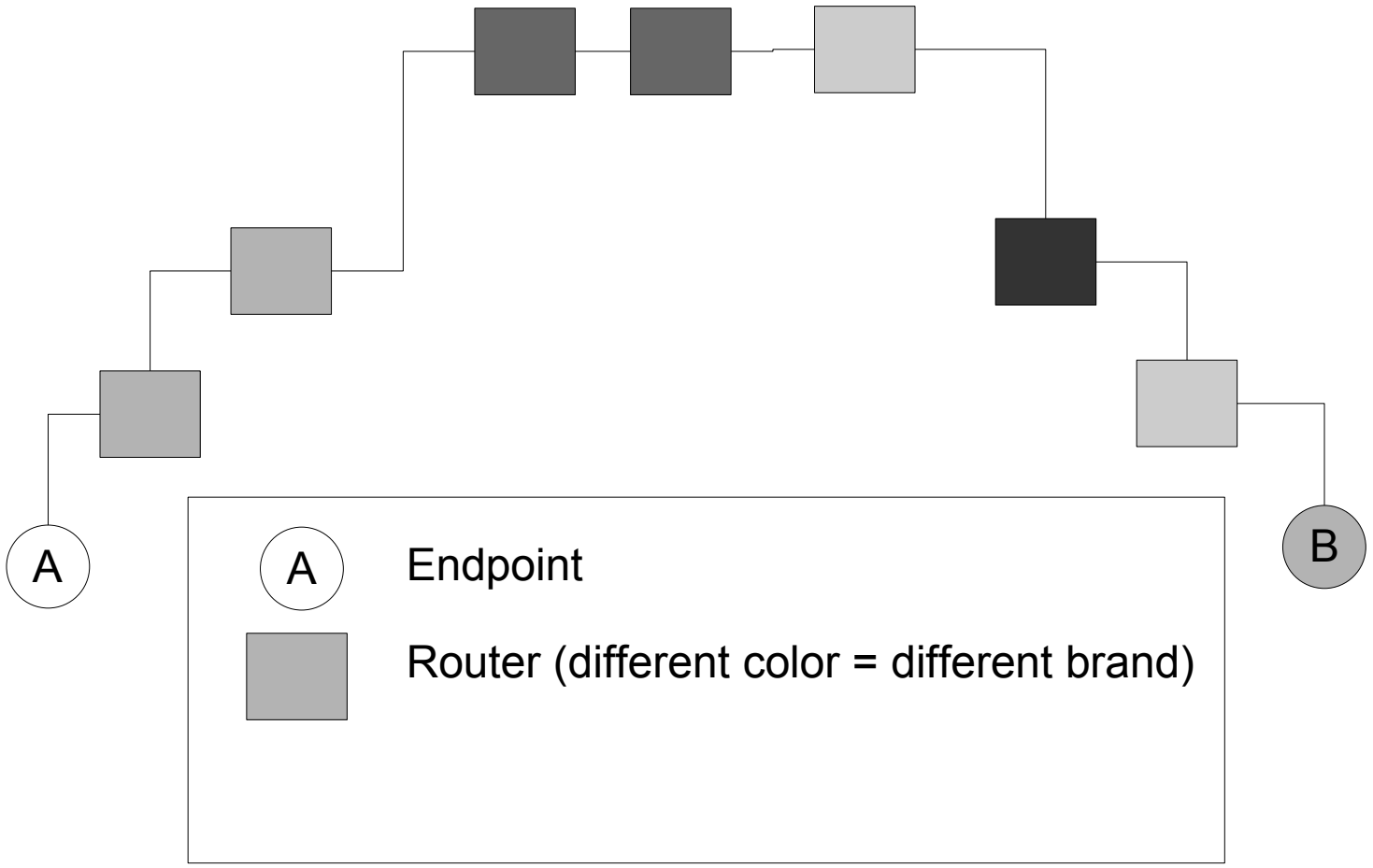
Possibilities

- You probably should not be thinking, “How can I extract info from Google” but rather:
  - “Who else in the world has tons of information that I'd like to get access to, but in a more structured form and machine readable and how can I convince them to put up a web service?”
  - “What information do I have in my lab that I'd like to make accessible to **programs** to use and how can I leverage that openness to attract funding?”
- This interface was designed 3+ years ago – how many changes has Google gone through since? -- This is programming while expecting change!

Example 3

- The amount of data that Atlas has to move around the world is almost beyond comprehension. And while having a network from point A to point B is necessary, it is not sufficient to permit the level of access needed – you don't want to be competing with teenagers sharing their entire collection of DVDs over a peer-to-peer network.
- We'd like to arrange for a guaranteed Quality of Service for network traffic. (Terapaths Project)

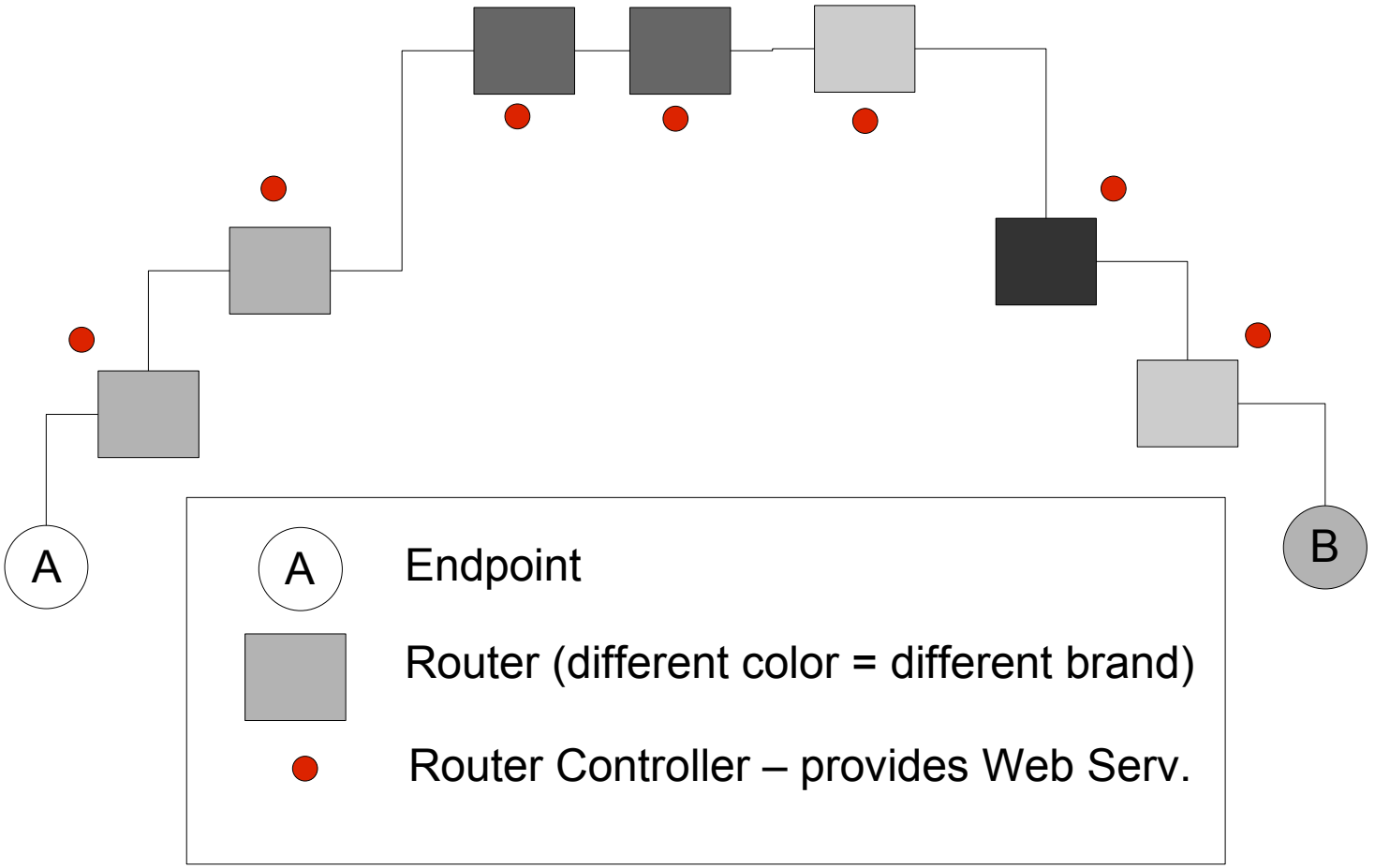
The Wire Layout
How to Program the Routers?



Problems & Features

- Problems
  - Every router is potentially different
  - Every router owner is distrustful (with good reason)
- Features
  - Everyone wants to solve this problem
  - All routers are programmable
  - “What” has to be done is known – treat certain network packets in a special way. “How to do it varies”
- This is begging for an “interface” definition

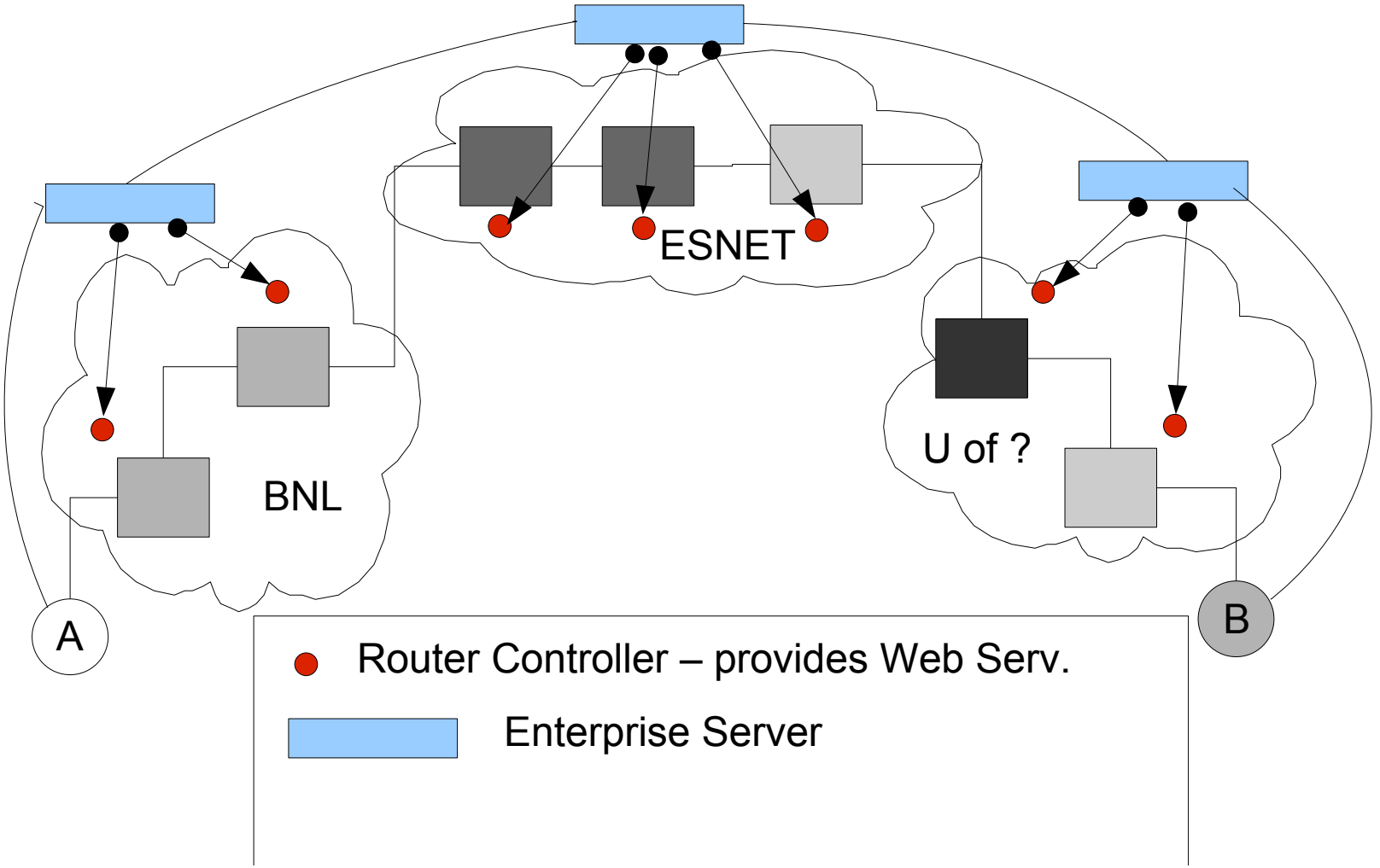
So every router has some intelligent entity that can control it...



More Problems

- There are still big problems
  - To program the routers, we would have to know the internal structure of the networks
  - Nobody is going to let us talk directly to their router
  - How to coordinate the many other hosts that use all or part of this path?
- So – add an “enterprise” layer – that can program (with the proper request, authorization etc.) a number of routers – moving data through the “cloud” and work out conflicts involving multiple requests.

Final Architecture



Advantages

- We've been able to develop a (gulp) framework
  - Router “drivers” are saved in databases
  - Internal router configurations are also placed in databases
  - This permits newcomers to easily incorporate themselves into this system – sometimes by only modifying the contents of databases
- We have developed a negotiation scheme (based on buying tickets on the web) that is quite useful for finding openings.

Does This Work in Practice?

- So far, yes
  - Esnet has adopted our enterprise interface (web services)
  - Other sites are going along as well (less programming for them), providing diverse environments to test this and forcing us to adopt a framework state of mind.
- This has got to be easier than Google's job!

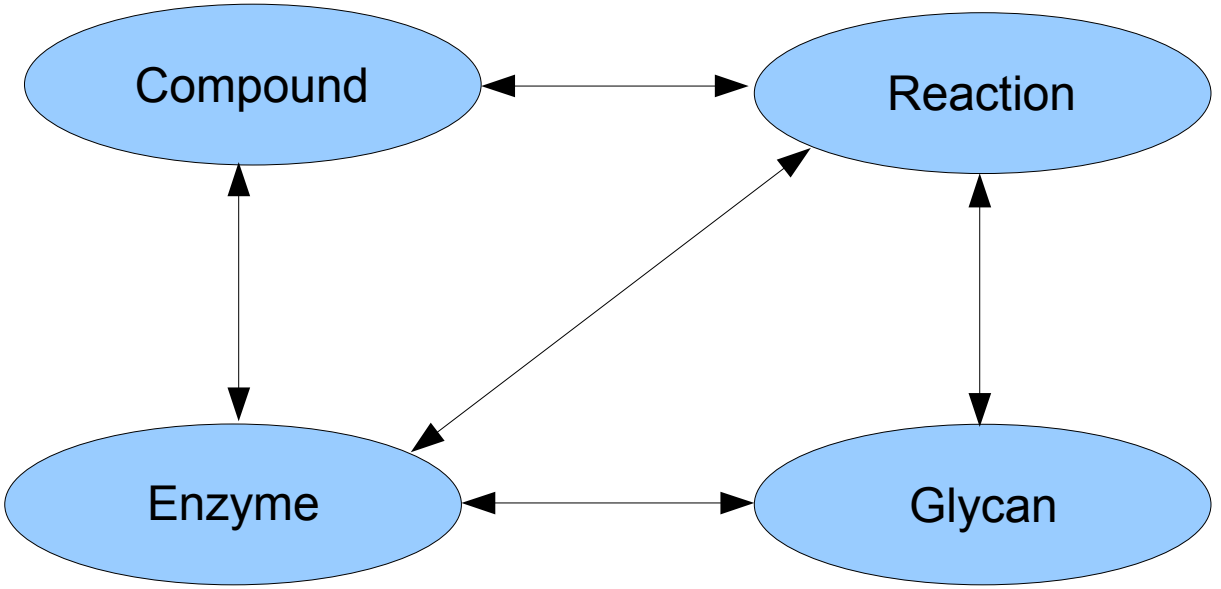
Lessons Learned

- You don't have to be controlling routers – any programmable device is a target...
  - Labs (microscopes, beam lines, lasers, etc.)
  - Programmers can focus on their area of expertise & their best language.
    - Device control (Matlab?)
    - Scheduling (Java?)
    - User Interface (php?)
    - Processing (C++, Java?)
  - Security (https, X509) is relatively painless
  - The programming sandbox is pretty secure as well

Final Example – KEGG

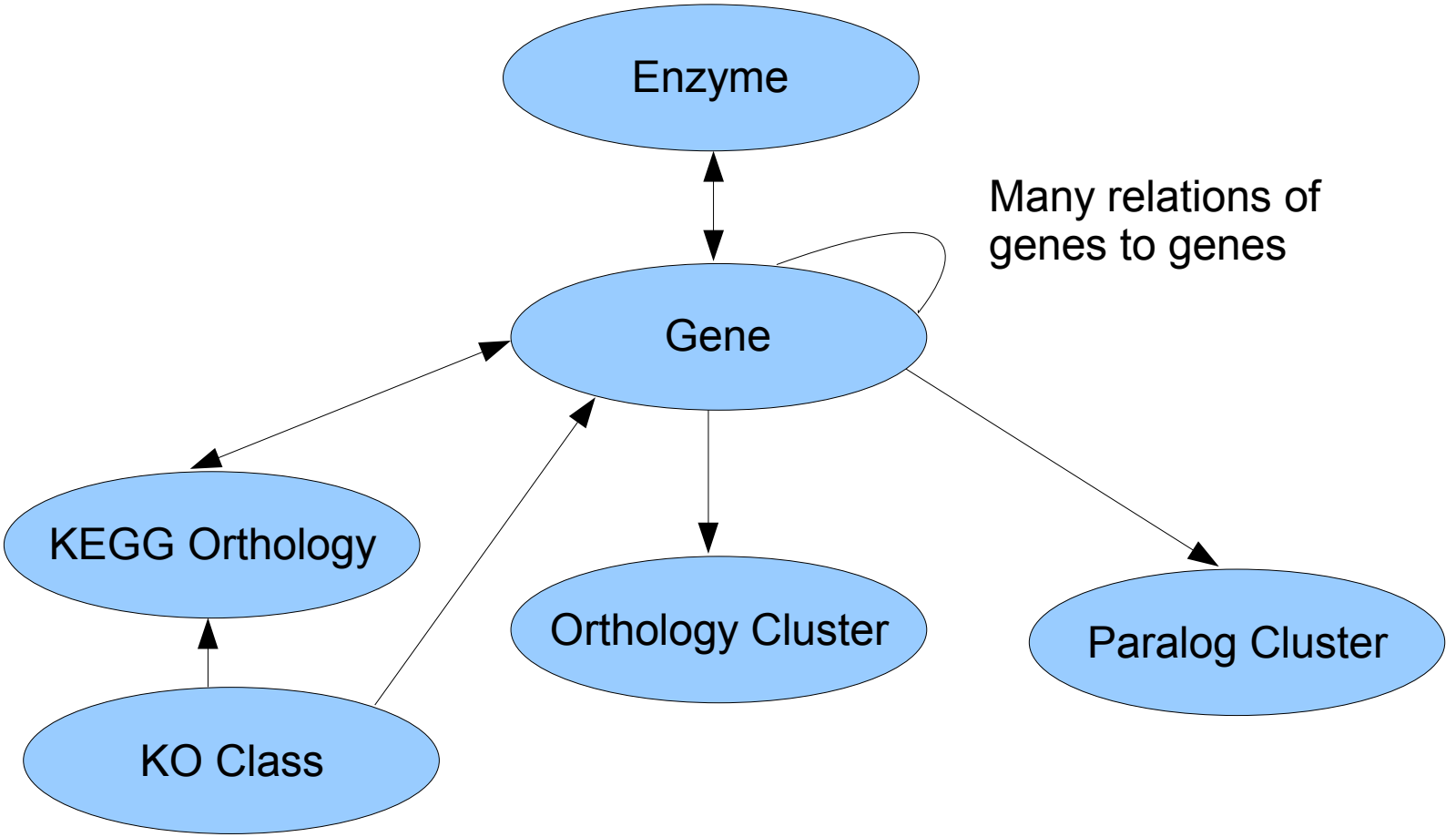
- Kyoto Encyclopedia of Genes and Genomic (http://www.genome.jp/kegg)
  - KEGG is a “suite” of databases including GENES, SSDB, PATHWAY, LIGAND, LinkDB, etc.
  - One of KEGGs primary accomplishments is to link together these databases, resolving some naming issues, keys, etc.
  - KEGG provides access to their work via the web or (of course) by Web Services – in particular, they provide a WSDL as well as handy Perl, Ruby, Python, and Java libraries.

A Pedestrian View of KEGG's Data Organization -1



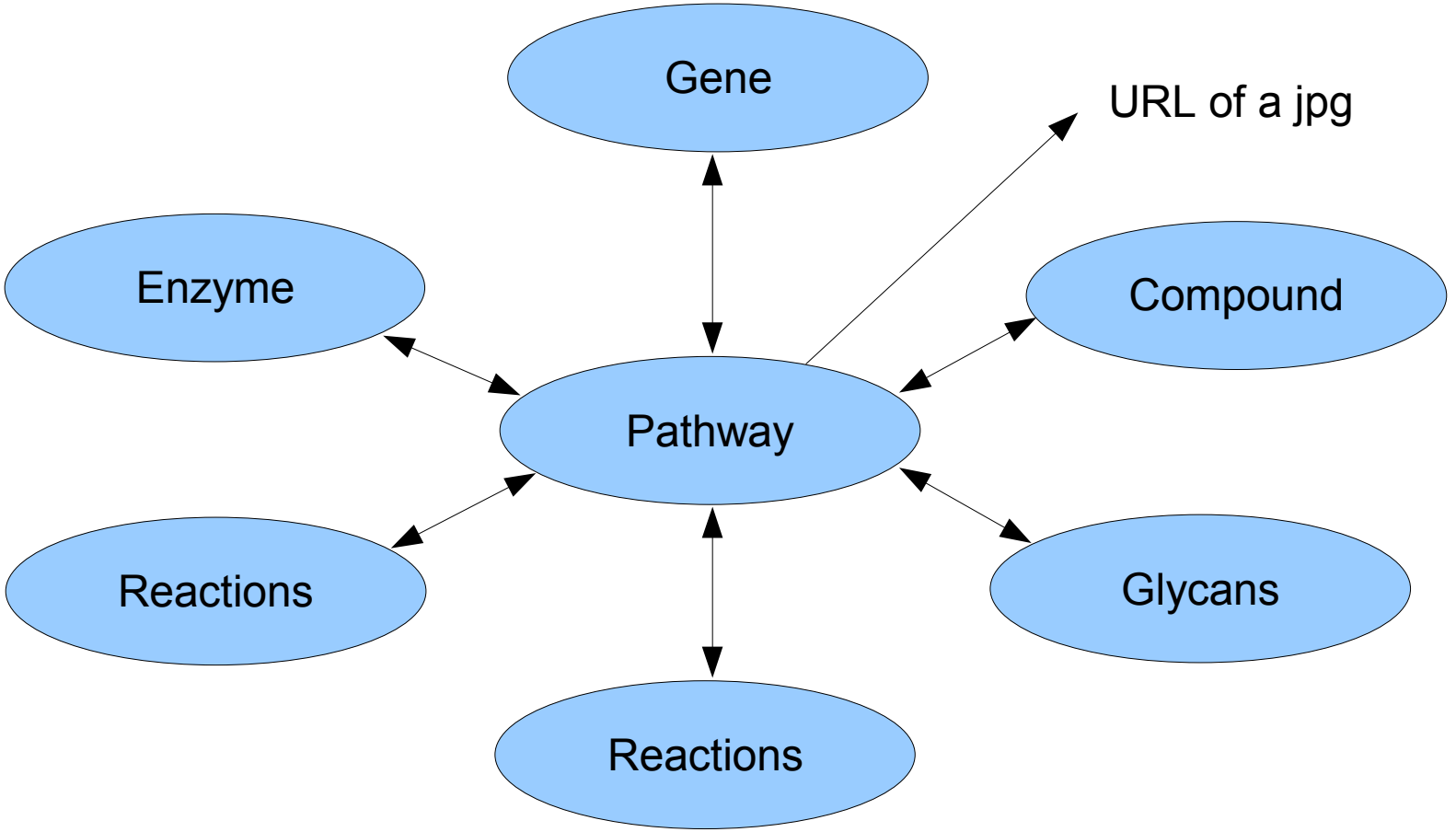
Every double headed arrow represents 2 web services – knowing the id of a member of one set, gives you a list of related members of another set.

A Pedestrian View of KEGG's Data Model - 2



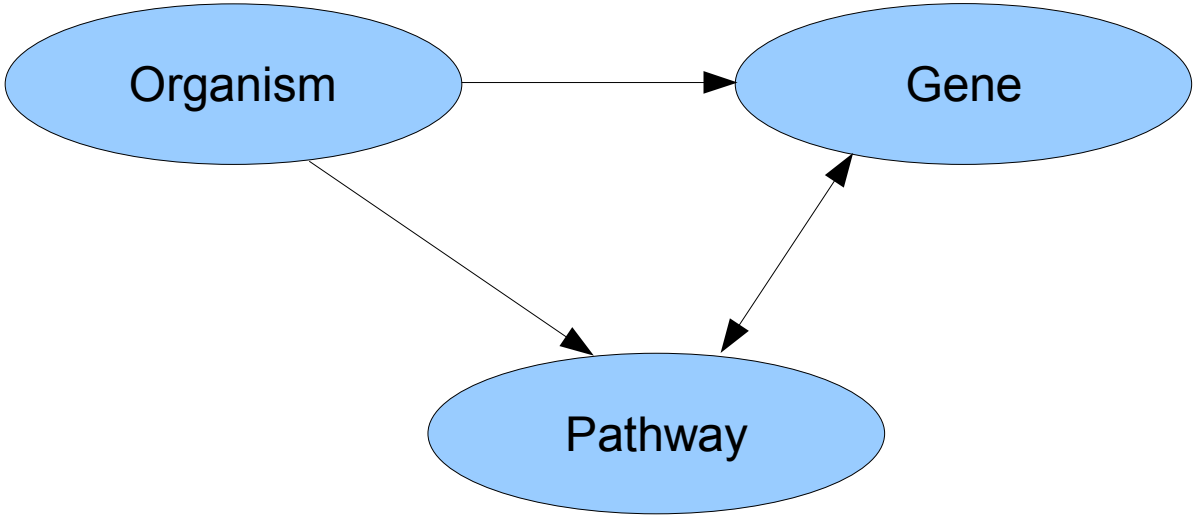
Each line once again represents one or more web services to map from one set to another.

A Pedestrian View of KEGG's Data Model 3

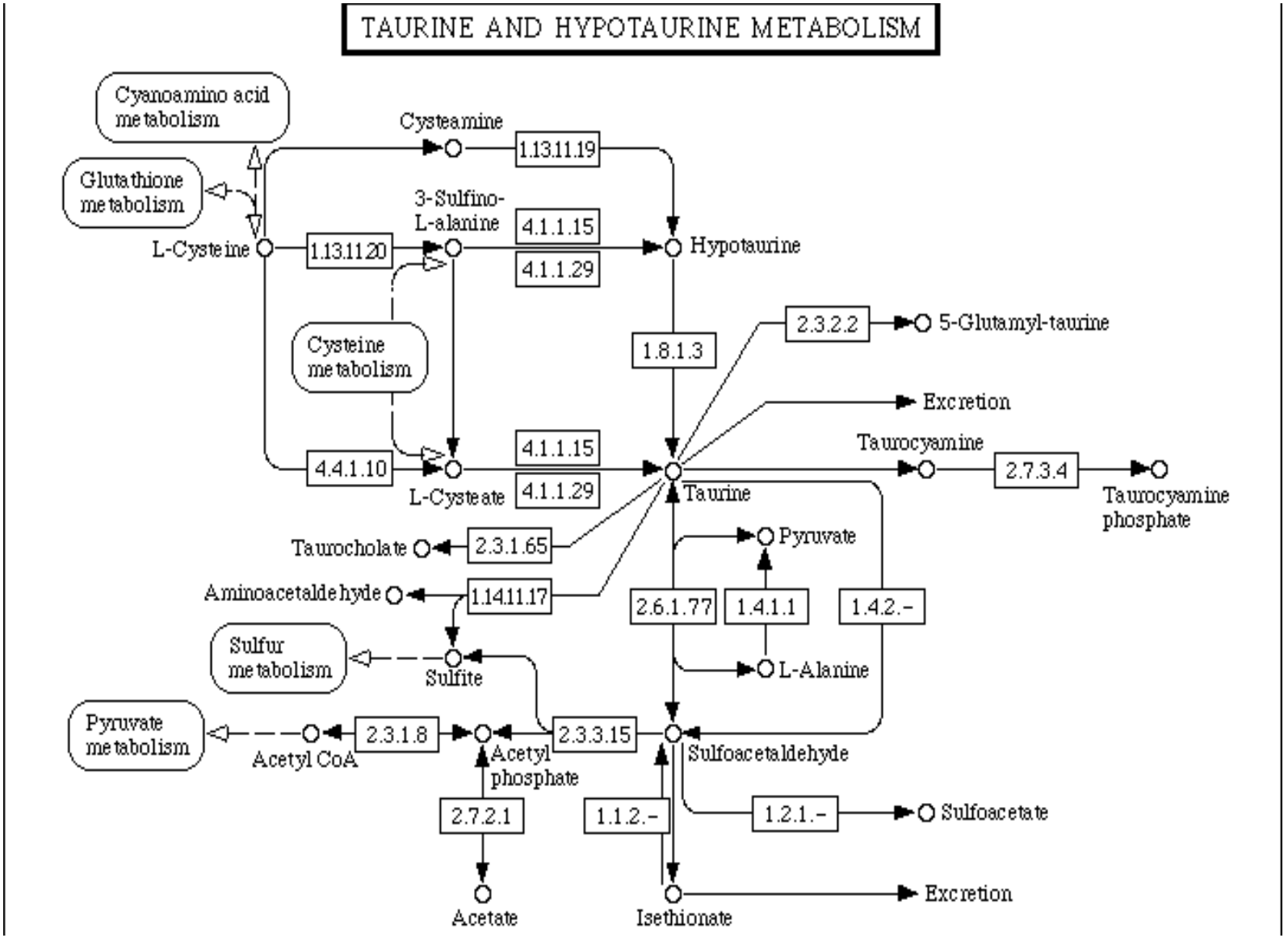


Again, web services everywhere

And Finally...



And the Pathway Diagrams?



The Pathway Diagrams

- The Pathway diagrams are hand drawn, but via web services,
  - one can extract virtually all of the information in a computer friendly fashion
  - One can modify the presentation of the pathway diagram, illuminating certain paths or components through the use of color.

Some Notes about KEGG...

- All of the above diagrams are maps between “ids”. For example a reaction-id to one or more enzyme-ids. Other methods in the web services provide access to the appropriate database from the id itself.
- What this defines, is a “semantic network” - a data structure that is set up for data mining, spidering, and some interesting AI type algorithms.
- In general, it was neither the best nor worst way to do this – but it begs for a more investigation

So where to from here?

- Programmers
  - Learn to design programs by using interfaces – it will help you design locally as well as preparing your code for the future.
  - Don't get stuck in the XML/SOAP mud use tools that make them disappear.
  - By learning Web Services technology, you are preparing for “Grid Services”, future Microsoft technology (NY Times, March 21, 2006) and what is considered to be the hottest programming area today.

So, where to from here?

- To Designers
  - Web Services are real – in business and science
  - If you are inventing new ways to provide services remotely, **STOP.** Consider the use of Web Services.
  - In proposals you should propose to both use and provide web services. While the services may be substantial, even tenuous links between two different objects may lead to new understandings (see web surfing).
  - Look for opportunities to leverage other entity's strengths by making it seem as if they were located at BNL, and vice versa. (e.g. If BlueGene becomes a reality here, could we be a computing arm of KEGG?)

Conclusion

It is worth keeping in mind the Faber-Castell TR3 Calculator. With a browser only interface to information, you are using the slide rule side of the instrument. Nice to hold, delightful to estimate the value beneath the cursor and a pleasure to demonstrate one's skill with the slide rule.

But with a Web Services interface on the reverse side, you have all of the advantages of computerized processing as well. Speed, accuracy and the untiring ability to search for links amid the data.